18 n
0300

00169.002262

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)	
	:	Examiner: Not Yet Assigned
KHAN PHI VAN	)	
	:	Group Art Unit: Not Yet Assigned
Application No.: 10/022,376	)	
	:	
Filed: December 20, 2001	)	
	:	
For: RENDERING OBJECTS	)	February 21, 2002

Commissioner for Patents
Washington, D.C. 20231

SUBMISSION OF PRIORITY DOCUMENTS

Sir:

In support of Applicant's claim for priority under 35 U.S.C. § 119, enclosed are certified copies of the following foreign applications:

PR 2312, filed December 22, 2000; and

PR 4318, filed April 10, 2001.

Applicant's undersigned attorney may be reached in our New York office by telephone at (212) 218-2100. All correspondence should continue to be directed to our address given below.

Respectfully submitted,

Carl B. Wischhusen
Attorney for Applicant

Registration No. 43,279

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3801
Facsimile: (212) 218-2200

NY_MAIN 237212v1

AUSTRALIA

Patent Office
Canberra

CERTIFIED COPY OF
PRIORITY DOCUMENT

I, GAYE TURNER, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PR 2312 for a patent by CANON KABUSHIKI KAISHA filed on 22 December 2000.

WITNESS my hand this
Second day of January 2002

GAYE TURNER
TEAM LEADER EXAMINATION
SUPPORT AND SALES

COMMONWEALTH OF AUSTRALIA
AUSTRALIA
PATENT OFFICE

S&F Ref: 529343

ORIGINAL

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION FOR THE INVENTION ENTITLED:

Rendering Objects

Name and Address of Applicant:

Canon Kabushiki Kaisha, incorporated in Japan, of 30-2, Shimomaruko 3-chome, Ohta-ku, Tokyo, 146, Japan

Name of Inventor:

Khanh Phi Van Doan

This invention is best described in the following statement:

RENDERING OBJECTS

Technical Field of the Invention

The present invention relates generally to rendering objects and, in particular, to a method and apparatus for rendering objects. The invention also relates to a computer readable medium comprising a computer program for rendering objects.

Background Art

Scan conversion is a typical computer graphics process in which geometrical objects such as lines and polygons are converted into pixel data for displaying on a raster device. Scan conversion can either operate in a simple aliased or more sophisticated anti-aliased mode.

In aliased mode, each display pixel is assigned one of two values according to whether it is classified as being inside or outside of the object being scan converted. It is well known that in this mode, aliasing effects can occur along the edges of objects which give rise to "jagged" appearance. Techniques aimed at reducing or eliminating this effect are referred to as anti-aliasing techniques. They work by blending the object's colour with the colour of the background along pixels lying on the edges of the object to create smooth transitions between pixels lying outside and inside the object. Scan conversion methods that incorporate these techniques are said to be operating in anti-aliased mode.

Existing anti-aliasing techniques generally fall into one of three types: filtering, multi-point sampling, and area sampling. Filtering techniques work by applying a low-pass filter to the pixel values produced by an aliased scan conversion of an object to remove the high spatial frequency components that give rise to the "jagged" appearance of edges. Since this requires a matrix multiplication at each pixel, filtering techniques are computationally expensive. Filtering also has the effect of blurring horizontal and vertical edges that fall exactly on pixel boundaries that would otherwise appear sharp when scan converted using other anti-aliasing techniques, which may be undesirable.

In multi-point sampling, each pixel is sampled at several different locations at which tests are made to identify which of these points lie inside the object being scan converted. The pixel is then assigned a value based on the number of such points. Variations of this technique also exist where sampling is performed along several

continuous horizontal or vertical line segments within each pixel instead of at discrete locations. The pixel is assigned a value based on the total length of these line segments that lie inside the object being scan converted.

A disadvantage of multi-point sampling is that it does not handle thin edges well. Figs. 1 and 2 illustrate the problems associated with multi-point sampling. The object being scan converted in both Figs 1 and 2 is a thin, near-horizontal line 10. In Fig. 1, sampling is performed at nine equally spaced points within each pixel 1, 2, 3, 4, 5, 6, 7, and 8. The white and black circles represent sampling points that lie outside and inside of the line 10 respectively. It can be seen that since all sampling points in pixels 3 and 6 lie outside of the line, these pixels will be treated as if they lie completely outside of the line, and hence the scan converted line will appear broken. Namely, the line 10 is represented by pixels 1, 2, 4, 5, 7, and 8.

A similar problem exists in Fig. 2, where sampling is performed over three horizontal line segments per pixel denoted by the dotted lines. The portions of these line segments that lie inside the line being scan converted is highlighted black in the figure. Again, the scan converted line will appear broken at pixels 3 and 6.

The third type of anti-aliasing technique is area sampling, where each pixel is assigned a value according to the percentage of the pixel area that falls inside the object being drawn. Its advantage over multi-point sampling is that it does not suffer from the problem illustrated in Figs 1 and 2 that arise when scan converting thin lines. The overriding disadvantage of area sampling is that whilst straight lines and simple polygons can easily be handled, complex, self-overlapping (or self-intersecting) polygons pose a problem since exact area calculations are too computationally intensive to be performed on a per pixel basis.

Although techniques are available to convert complex, self-overlapping polygons into one or more simple, non-intersecting polygons, the computational effort required make them impractical.

When scan converting a complex or self-overlapping polygon, it is necessary to select a fill rule. A polygon is a set of one or more closed curves each comprising of a number of vertices connected by straight line segments. Each closed curve is also known

as a contour and has an associated direction. A polygon is said to be *simple* if it comprises of only a single contour, otherwise it is said to be *complex*. A polygon is also said to be *self-overlapping* or *self-intersecting* if one or more of its contours crosses over itself or over other contours. Examples of the possible different types of polygons are shown in Figs. 3(A), 3(B), 3(C), and 3(D).

Odd-even and non-zero winding are two fill rules well known to those skilled in the art based on the winding count of a point. The winding count of a point is defined as follows: draw an arbitrary path from any point outside of the polygon to this point. Count the number of times a contour crosses the path from one side of the path to the other, and the number of times a contour crosses the path in the opposite direction. The winding count of the point is the obtained by subtracting the second number from the first.

When using the odd-even fill rule, points that have odd winding counts are considered to be inside the polygon and are hence filled with the polygon's colour and opacity, whilst points with even winding counts are considered to be outside the polygon and are not filled.

When using the non-zero winding fill rule, points that have zero winding counts are considered to be outside the polygon, whilst points with non-zero winding counts are considered to be inside the polygon.

The publication United States Patent 6084596 discloses a third fill rule, called "winding-counting". Unlike the non-zero winding fill rules, in which all pixels classified as being inside the polygon are rendered with uniform colour and opacity, winding-counting assigns an opacity value to a pixel according to its absolute winding count. More specifically, pixels with a zero winding count are classified as being outside the polygon and are not filled, and pixels with a +1 or –1 winding count are filled in the same manner as in the odd-even and non-zero winding rule.

The pixels with a winding count of n, where $|n| > 1$, are rendered by performing $|n|$ repeated rendering operations, each time using a pixel with the colour and opacity of the polygon. For fully opaque polygons, this produces identical results to the non-zero winding fill rule. For partially transparent, complex and/or self-overlapping polygons

however, winding-counting gives the effect of the polygons being made up of several overlapping layers.

A comparison between the three fill rules is shown in Figs 4(a), 4(b), and Fig. 4(c). In Fig. 4(a), a partially transparent polygon is rendered using the odd-even fill rule. In (b) and (c), the same polygon is rendered using the non-zero winding and the winding-counting fill rules respectively. The centre region of Fig. 4(c) has a winding count of 2 and hence has been drawn with a higher opacity than the surrounding region.

However, the scan conversion method of the publication United States Patent 6084596 suffers from "aliasing" effects.

The reference to the publication cited in the "Background Art" section herein is not to be taken as an admission that the publication constitutes common general knowledge .

Summary of the Invention

It is an object of the present invention to substantially overcome, or at least ameliorate, one or more disadvantages of existing arrangements.

According to a first aspect of the invention, there is provided a method of rendering objects using a scanline process, the method comprising, for each of said object within a scanline, the steps of: determining each boundary pixel that overlaps both sides of the border of the object; computing a real opacity of each said boundary pixel, wherein said real opacity is representative of the sum of real opacities of respective subregions of the pixel, wherein said real opacity of a said subregion is dependent upon an intrinsic opacity of said object and a winding count for that subregion, wherein said winding count is incremented for directional border crossings in a first direction and is decremented for said directional border crossings in a second direction; and rendering each said boundary pixel with said determined real opacity.

According to another aspect of the invention, there is provided an apparatus for implementing any one of the aforementioned methods.

According to another aspect of the invention there is provided a computer readable medium comprising a computer program for implementing any one of the methods described above.

Other aspects of the invention are also disclosed.

Brief Description of the Drawings

A number of embodiments of the present invention will now be described with reference to the drawings, in which:

Fig. 1 illustrates the results of a prior art anti-aliasing method using multi-point sampling;

Fig. 2 illustrates the results of a prior art anti-aliasing method using line segment sampling;

Figs. 3(A), 3(B), 3(C), and 3(D) illustrate different types of objects;

Fig. 4(A), 4(B), and 4(C) illustrate the results of three different fill rules;

Fig. 5 illustrates the computing of real opacity of a pixel in accordance with a winding counting rule;

Fig. 6 illustrates the computing of real opacity of a pixel in accordance with a winding counting rule;

Fig. 7 is a method of rendering objects in accordance with a first arrangement;

Fig. 8 is a schematic block diagram of a general purpose computer upon which arrangements described can be practiced.

Detailed Description including Best Mode

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have for the purposes of this description the same function(s) or operation(s), unless the contrary intention appears.

A formal definition of a winding-counting fill rule is presented that will enable anti-aliased scan conversion methods to be devised unambiguously.

Consider the case where two simple, partially transparent polygons of opacity α are composited together using the "*over*" operator as defined in "Compositing Digital Images" by Porter and Duff, SIGGRAPH 84, 1984 (incorporated herein by reference). According to the definition of this operator, when a point with an opacity of α_L is composited "*over*" another point of opacity α_R, the resulting opacity is given by

$$\alpha_L + \alpha_R - \alpha_L\alpha_R$$

In the present example, since the opacity of both polygons is α, the opacity of the intersection region between the two polygons is

$$2\alpha - \alpha^2$$

Now consider the case where there are n overlapping polygons instead of 2. Again each polygon is partially transparent with an opacity of α. When these polygons are composited together using the "*over*" operator as defined in Porter and Duff, it can be shown by mathematical induction that the resulting opacity of the intersection region between all n polygons is given by

$$1 - (1 - \alpha)^n$$

Since the aim of the winding-counting fill rule is to create the effect of polygons being made up of several layers in regions where the absolute winding counts are greater than 1, it is desirable that such regions are rendered with the same opacity as that would be obtained by compositing together the same number of such layers.

The winding-counting fill rule is thus defined as follows:

If the *intrinsic opacity* of an object at a point P is α, and P has a winding count of n, then point P is assigned a *real opacity* value of

$$1 - (1 - \alpha)^{|n|} \qquad \text{- (1)}$$

where the opacity α is an intrinsic property of point P

Under the winding-counting fill rule, objects are rendered according to their real opacities. In contrast, objects are rendered according to their intrinsic opacities under the non-zero winding fill rule.

For pixels with uniform winding counts, the above definition produces identical results to that given in Porter and Duff. The new definition however, allows objects to be rendered more efficiently, since a pixel with a winding count of n needs to be composited only once rather than $|n|$ times as described in United States Patent 6084596. Although Eq (1) needs to be evaluated to determine the real opacity from the winding count of the pixel, its computational cost can usually be amortised over many pixels with the same winding counts and hence will likely be insignificant. An example of how this can be achieved is as follows:

1. Given an object with uniform intrinsic opacity to be rendered, pre-compute the real opacities for points with absolute winding counts from 1 to m, where m is some positive integer. Store these in a look up table indexed by the absolute winding count.
2. Perform the scan conversion as described in United States Patent 6084596, but instead of compositing a pixel whose winding count is n $|n|$ times, do the following

   2a. If $|n| \leq m$, then look up the table created in Step 1 to obtain the real opacity associated with a winding count of $|n|$.

   2b. Otherwise compute the real opacity using Eq (1).

   2c. Render the pixel once with the real opacity value obtained in Step 2a or 2b.

The above method takes advantage of the fact that for most cases, the maximum absolute winding count over the entire object is usually rather low, and hence the pre-computed look up table will likely cover the majority of pixels.

The new definition allows existing anti-aliased scan conversion techniques to be extended to support the winding-counting fill rule. For example, in multi-point sampling methods, instead of rendering each pixel based on the total number of sampling points that lie inside the polygon, each pixel is rendered based on the sum of the real opacities at all sampling points. This is illustrated in Fig.5, which shows a pixel comprising a 4x4 array of sampling points.

In the Fig. 5, the black circles represents sampling points, whilst the numbers shown indicate the winding counts of the four subregions of the pixel. Since the real opacities associated with winding counts of 0, 1, and 2 are 0, α, and $2\alpha - \alpha^2$ respectively, where α is the intrinsic opacity, the sum of the real opacities at all sampling points is

$$6\alpha + 5(2\alpha - \alpha^2)$$

The pixel is then assigned an opacity value equal to this sum divided by the total number of sampling points.

Anti-aliased scan conversion methods based on area sampling can be extended to support the winding-counting fill rule in a similar way. Instead of rendering each pixel according to the percentage area of the pixel that falls inside the polygon, it is rendered according to the sum of the percentage areas of the different subregions that make up the pixel, weighted by their real opacities. This is illustrated in the example shown in Fig.6,

which shows the four sub-regions of the polygon within the pixel. For the example shown in Fig. 6 this sum is computed to be

$$\alpha(x+y)+(2\alpha-\alpha^2)z$$

where x,y are the percentage areas of the subregions whose winding count is 1, and z is the percentage area of the sub-region whose winding count is 2.

Turning now to Fig. 7, there is shown a method 700 of rendering objects using a scanline process. The method 700 comprises the following steps performed for each one of the objects within each scanline.

The method 700 commences at step 701, where any necessary parameters are initialised. The method 700 may be called by a main method for the processing of each object within each scanline. After completion of step 701, the method 700 continues to step 702.

During step 702, the method 700 determines the intersections of all the edges of the object that intersect the current scanline. Preferably, these intersections are specified by the x co-ordinate location that the edge intersects with the top of the current scanline. Alternatively, these intersections may be specified by the x co-ordinate location that the edge intersects with the bottom or any horizontal line that runs through the current scanline. After completion of step 702, the method 700 proceeds to step 704.

During step 704, the method determines those "boundary" pixels which straddle the edges of the object within the current scanline. The intersections determined during step 702 specify the locations within the boundary pixel where the edge of the object intersects the current scanline. Steps 704 and 702 may interchanged in the sequence of steps of the method or performed simultaneously together.

After completion of step 704, the method proceeds to step 706, where the method 700 determines for each pixel in the current scanline a directional count of the number of edges of the object that intersect the current scanline prior to the current pixel. The directional count sums the aforementioned edges, which have a first direction and sums the aforesaid edges, which have a second direction and subtracts the first sum from the second to obtain the count. The directional count is in effect a winding count of the current pixel or a sub-region of the current pixel, depending upon whether an edge of the

object intersects the previous pixel. After completion of step 706, the method proceeds to decision block 708.

The method 700 scans each pixel with the current scanline and checks in decision block 708 whether the currently scanned pixel is a boundary pixel.

If the decision block 708 returns TRUE (YES) the method proceeds to step 710 where the real opacity of the boundary pixel is computed. The method 710 computes the real opacity of the boundary pixel in accordance with the technique described with reference to Fig. 5 or 6. The directional count of the previous pixel is used as a basis for determining the winding count of the different regions or sampling points within the current pixel. After completion of step 710, the method proceeds to step 714.

On the other hand, if the decision block 708 returns FALSE (NO) the method proceeds to step 712 where the real opacity of the non boundary pixel is computed. The method 710 computes the real opacity of the current pixel according to formulae (1), wherein the winding count of the current pixel is the directional count of the previous pixel.

After completion of either steps 710 or 712, the method 700 renders the pixel in accordance with the computed real opacity.

The method of rendering objects is preferably practiced using a general-purpose computer system 700, such as that shown in Fig. 8 wherein the processes of Figs.7, 6 or 7 may be implemented as software, such as an application program executing within the computer system 700. In particular, the steps of method of rendering objects are effected by instructions in the software that are carried out by the computer. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer preferably effects an advantageous apparatus for rendering objects.

The computer system 700 comprises a computer module 701, input devices such as a keyboard 702 and mouse 703, output devices including a printer 715 and a display

device 714. A Modulator-Demodulator (Modem) transceiver device 716 is used by the computer module 701 for communicating to and from a communications network 720, for example connectable via a telephone line 721 or other functional medium. The modem 716 can be used to obtain access to the Internet, and other network systems, such as a Local Area Network (LAN) or a Wide Area Network (WAN).

The computer module 701 typically includes at least one processor unit 705, a memory unit 706, for example formed from semiconductor random access memory (RAM) and read only memory (ROM), input/output (I/O) interfaces including a video interface 707, and an I/O interface 713 for the keyboard 702 and mouse 703 and optionally a joystick (not illustrated), and an interface 708 for the modem 716. A storage device 709 is provided and typically includes a hard disk drive 710 and a floppy disk drive 711. A magnetic tape drive (not illustrated) may also be used. A CD-ROM drive 712 is typically provided as a non-volatile source of data. The components 705 to 713 of the computer module 701, typically communicate via an interconnected bus 704 and in a manner which results in a conventional mode of operation of the computer system 700 known to those in the relevant art. Examples of computers on which the described arrangements can be practised include IBM-PC's and compatibles, Sun Sparcstations or alike computer systems evolved therefrom.

Typically, the application program is resident on the hard disk drive 710 and read and controlled in its execution by the processor 705. Intermediate storage of the program and any data fetched from the network 720 may be accomplished using the semiconductor memory 706, possibly in concert with the hard disk drive 710. In some instances, the application program may be supplied to the user encoded on a CD-ROM or floppy disk and read via the corresponding drive 712 or 711, or alternatively may be read by the user from the network 720 via the modem device 716. Still further, the software can also be loaded into the computer system 700 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer module 701 and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including email transmissions and information recorded on websites and the like. The foregoing is merely

exemplary of relevant computer readable mediums. Other computer readable media may alternately be used.

The method of rendering objects may alternatively be implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions of the rendering. Such dedicated hardware may include graphic processors, digital signal processors, or one or more microprocessors and associated memories.

Industrial Applicability

It is apparent from the above that the arrangements described are applicable to the computer graphics industries).

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiment(s) being illustrative and not restrictive.

(Australia Only) In the context of this specification, the word "comprising" means "including principally but not necessarily solely" or "having" or "including" and not "consisting only of". Variations of the word comprising, such as "comprise" and "comprises" have corresponding meanings.

The claims defining the invention are as follows:

1. A method of rendering objects using a scanline process, the method comprising, for each of said object within a scanline, the steps of:

determining each boundary pixel that overlaps both sides of the border of the object;

computing a real opacity of each said boundary pixel, wherein said real opacity is representative of the sum of real opacities of respective subregions of the pixel, wherein said real opacity of a said subregion is dependent upon an intrinsic opacity of said object and a wind counting for that subregion, wherein said winding count is incremented for directional border crossings in a first direction and is decremented for said directional border crossings in a second direction; and

rendering each said boundary pixel with said determined real opacity.

2. A method as claimed in claim 1, wherein said real opacity of a said subregion is $1-(1-\alpha)^{|n|}$, where α is the intrinsic opacity of the object and n is the winding count for the subregion.

3. A method as claimed in claim 1, wherein said computing step of the real opacity of each said boundary pixel, comprises the sub-steps of:

computing real opacities of a plurality of sampling points within each said boundary pixel, wherein said real opacity at a said sampling point is dependent upon the intrinsic opacity of said object and the winding count for that sampling point, and

determining the real opacity of each said boundary pixel, wherein the real opacity of a said boundary pixel is the sum of the computed real opacities at the sampling points of said boundary pixel divided by the total number of sampling points in the pixel.

4. A method as claimed in claim 1, wherein said computing step of the real opacity of each said boundary pixel, comprises the sub-steps of:

determining those areas within each said boundary pixel which have a constant winding count;

computing real opacities of a plurality of areas within each said boundary pixel, wherein said real opacity of a said area is dependent upon the intrinsic opacity of said object and the winding count for that area; and

determining the real opacity of each said boundary pixel, wherein the real opacity of a said boundary pixel is the sum of the product of percentage areas of pixel occupied by each area of said boundary pixel and its computed real opacity.

computed real opacities of the areas of said boundary pixel.

5. A method as claimed in claim 1, wherein at least one of said objects is a simple polygon.

6. A method as claimed in claim 1, wherein at least one of said objects is a self-overlapping polygon.

7. A method as claimed in claim 1, wherein for a given object of uniform intrinsic opacity, said method further comprises:

computing the real opacities for winding counts 1 to m, where m is a positive integer;

8. A method as claimed in claim 1, wherein the method further comprises the sub-steps:

determining pixels of said inside of said object other than said boundary pixels;

computing a real opacity of each said inner pixel, wherein said real opacity is dependent upon an intrinsic opacity of said object and winding count for that inner pixel; and

rendering each said inner with said determined real opacity.

9. Apparatus for implementing any one of the aforesaid methods as claimed in the preceding claims.

10. A computer readable medium comprising a computer program for implementing any one of the aforesaid methods as claimed in the preceding claims.

11. A method for rendering objects, the method substantially as described herein with reference to Figs. 5, 6, and 7 of the accompanying drawings.

12. Apparatus for rendering objects, the apparatus substantially as described herein with reference to Figs. 5, 6, 7 and 8 of the accompanying drawings.

13. A computer readable medium comprising a computer program for rendering objects, the computer program substantially as described herein with reference to Figs. 5, 6, 7 and 8 of the accompanying drawings.

DATED this Twenty-second Day of December 2000

CANON KABUSHIKI KAISHA

Patent Attorneys for the Applicant

SPRUSON&FERGUSON

Abstract

RENDERING OBJECTS

A method of rendering objects using a scanline process, the method comprising, for each object within a scanline, the steps of: determining (704) each boundary pixel that overlaps both sides of the border of the object; computing (712) a real opacity of each said boundary pixel, wherein said real opacity is representative of the sum of real opacities of respective subregions of the pixel, wherein said real opacity of a said subregion is dependent upon an intrinsic opacity of said object and a winding count for that subregion, wherein said winding count is incremented for directional border crossings in a first direction and is decremented for said directional border crossings in a second direction; and rendering (714) each said boundary pixel with said determined real opacity.

Fig. 5,6,7

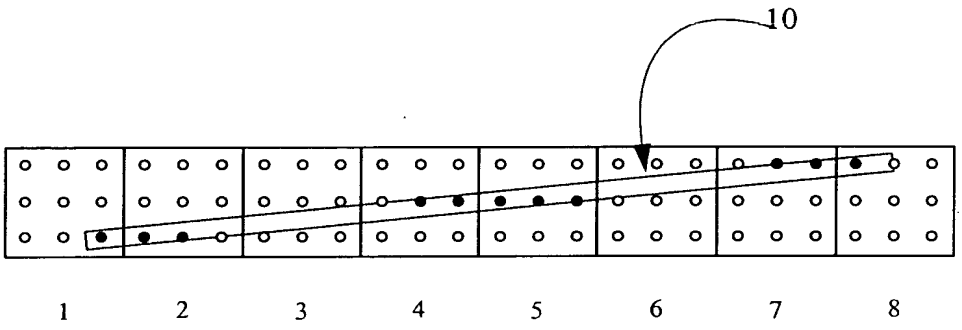


Fig. 1

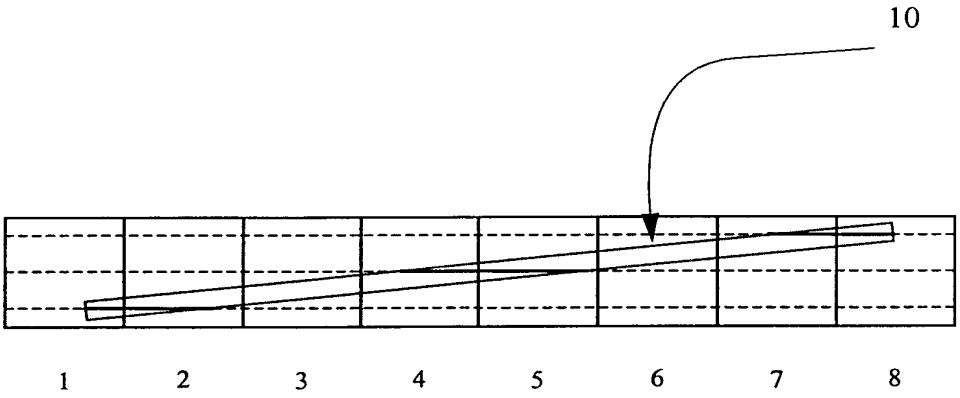
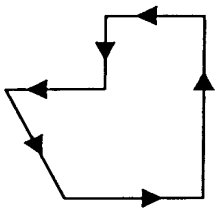


Fig. 2

Fig. 3(A)



a simple polygon

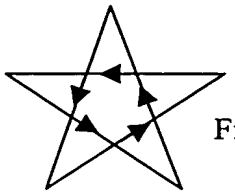
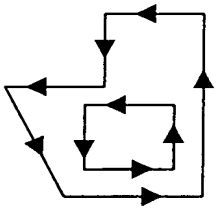


Fig. 3(B)

a simple, self-overlapping polygon

Fig. 3(C)



a complex polygon

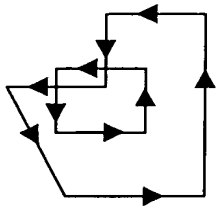
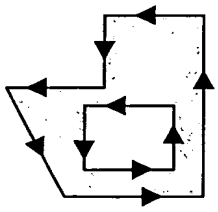


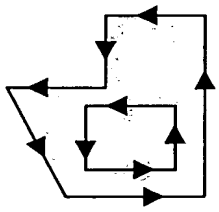
Fig. 3(D)

a complex, self-overlapping polygon



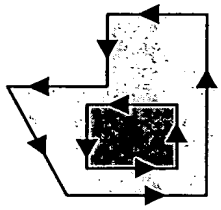
(a) odd-even fill rule

Fig. 4(A)



(b) non-zero winding fill rule

Fig. 4(B)



(c) winding-counting fill rule

Fig. 4(C)

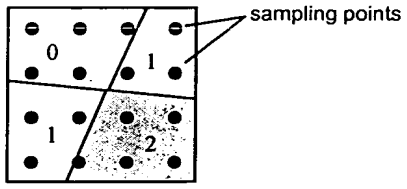


Fig. 5

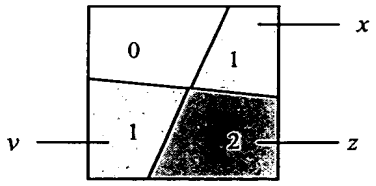


Fig. 6

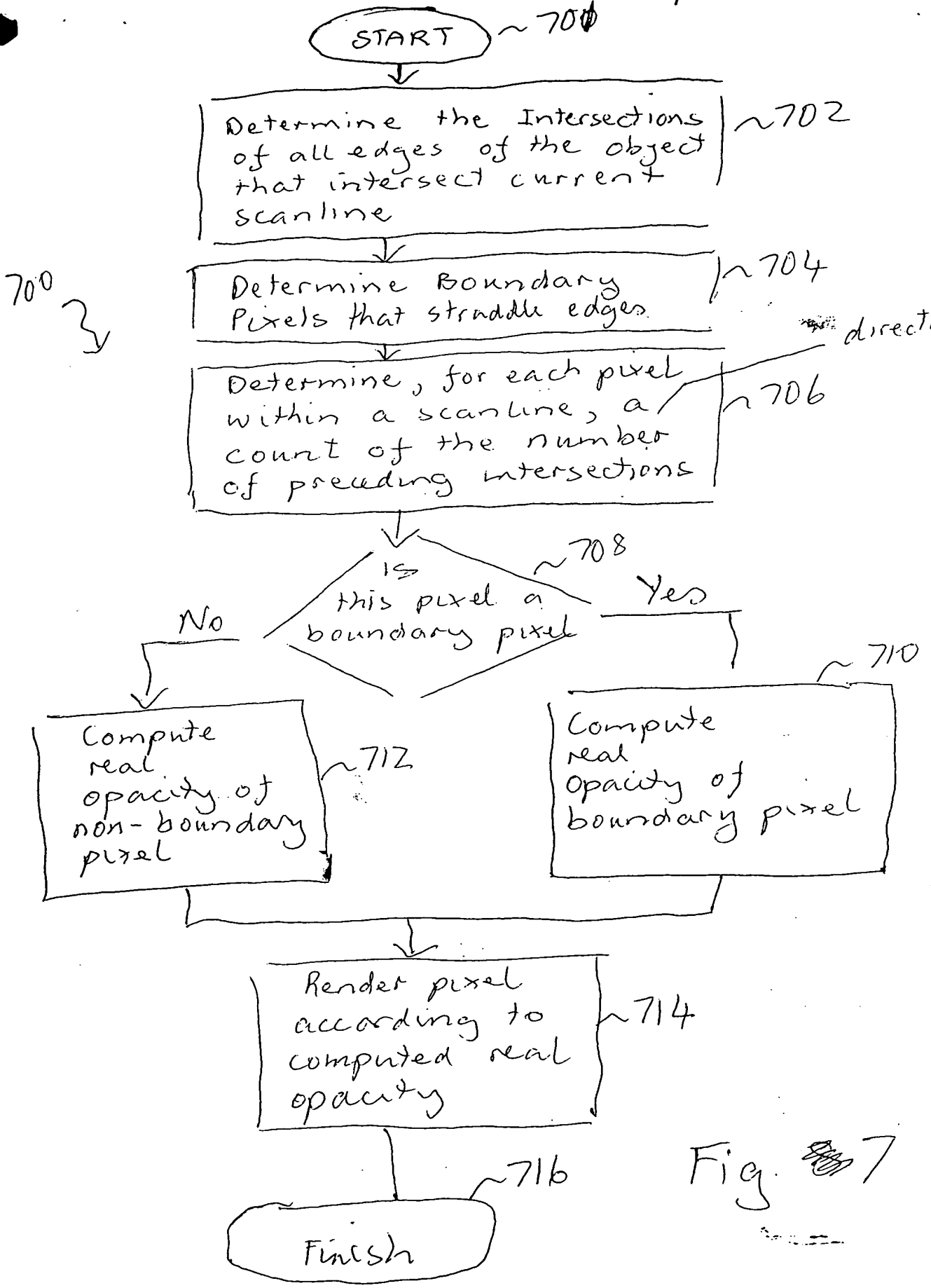

START
700
Determine the Intersections of all edges of the object that intersect current scanline
702
Determine Boundary Pixels that straddle edges
704
700
Determine, for each pixel within a scanline, a directional count of the number of preceding intersections
706
Is this pixel a boundary pixel
708
No
Yes
Compute real opacity of non-boundary pixel
712
Compute real opacity of boundary pixel
710
Render pixel according to computed real opacity
714
Finish
716

Fig. 7

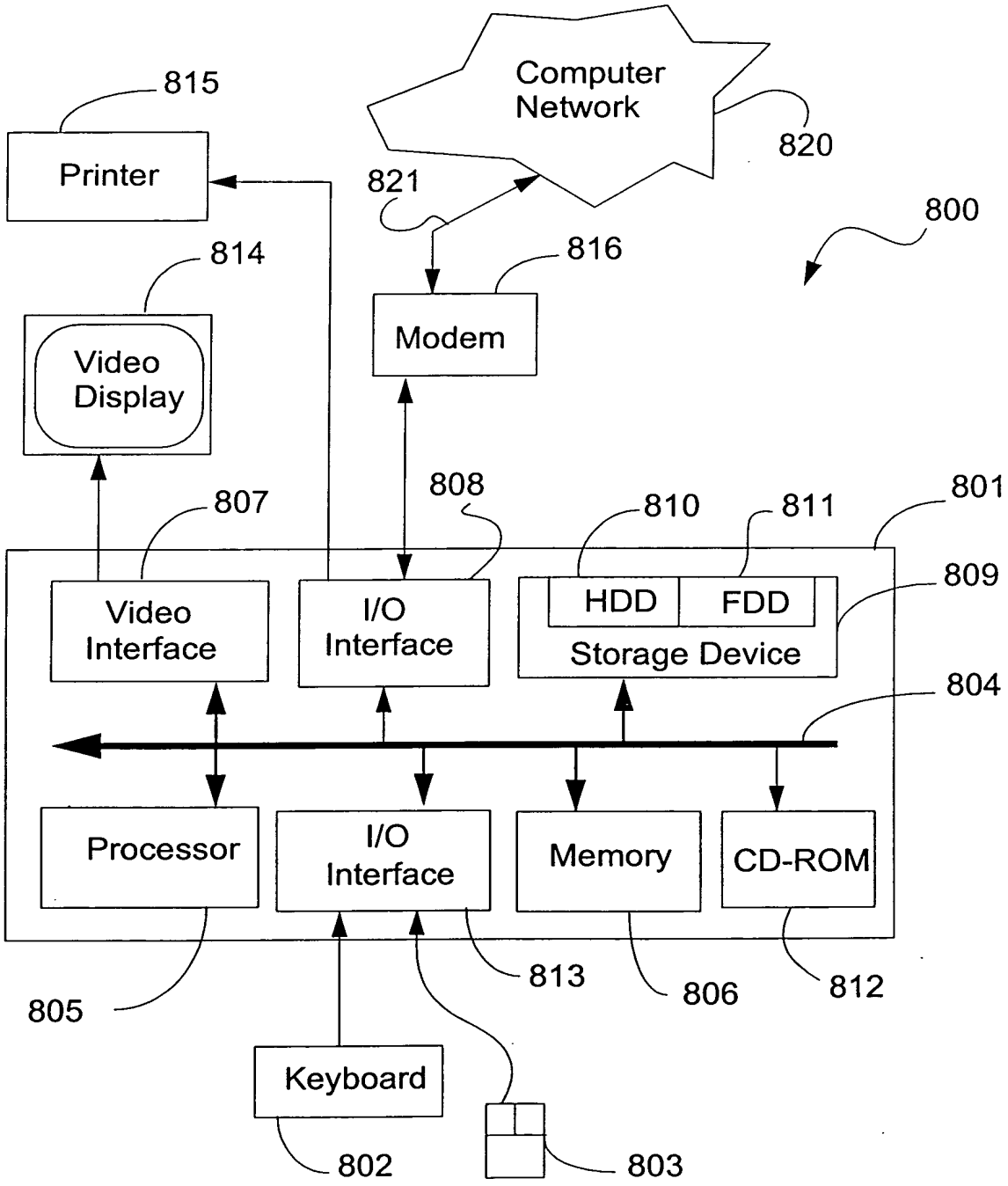

Computer Network
820
815
Printer
821
800
814
816
Modem
Video Display
808
807
810
811
801
Video Interface
I/O Interface
HDD
FDD
809
Storage Device
804
Processor
I/O Interface
Memory
CD-ROM
805
813
806
812
Keyboard
802
803

FIG. 8